XMine: A methodology for mining XML structure

Richi Nayak and Wina Iryadi

School of Information Systems, Queensland University of Technology
Brisbane, Australia {r.nayak@qut.edu.au}

Abstract. XML has become a standard for information exchange and retrieval on the Web. This paper presents the XMine methodology to group heterogeneous XML documents into separate meaningful classes by considering the linguistic and the hierarchical structure similarity. The empirical results demonstrate that the semantic and syntactic relationships and the path names context of elements play important role for producing good quality of clusters.

1 Introduction

The potential benefits of the rich semantics of XML have been recognized extensively for enhancing document handling over the vast amount of documents on the Web. The XML documents are usually associated with a schema definition that describes the structure of the document. A schema clustering process improves the document handling process by organising heterogeneous XML documents into groups based on structural similarity. Similarity of correspondence elements between XML documents is conducted efficiently using relevant XML schemas. We present a methodology, XMine, that quantitatively determines the similarity between heterogeneous XML schemas by considering the linguistic and the context of the elements as well as the hierarchical structure similarity, and groups them into separate classes.

Research on measuring the structural similarity and clustering of XML documents is gaining momentum [2,3,4]. XMine comes closer to a number of schema matching approaches based on schema only information such as XClust, Deep, Cupid, COMA, SF. However, XMine derives the structure similarity based on the maximal similar paths found by using the adapted sequential pattern mining algorithm [1]. Thus, this eliminates the element-to-element matching process, making XMine an efficient and accurate method. Lee et al. [3] also uses the sequential mining approach to quantify the structural similarity between XML documents. [3] defines the structural similarity only based on the 'ratio' between the maximal similar paths and the extracted paths of the base document. They do not include the element level hierarchy position, leading in erroneous match between two names occurring at two different positions or with different context. XMine overcomes this by including ***PNC***.

2 The XMine Methodology

The XMine methodology (figure 1) starts with the ***Structure Analyser*** that transforms the structure of a schema into a suitable tree model representation. This module performs simplification analysis of the schema trees in order to deal with nesting and repetition problems. The ***Element Analyser*** calculates the linguistic similarity of each

pair of element names based on the semantic and syntactic relationships. The semantic relationship (e.g. movie and film) is first measured for exploiting the similarity degree between the two token sets of names by looking up the *WordNet thesaurus* and *user-defined library* same as [4]. If there is no common elements are identified, the syntactic relationship (e.g. ctitle and title) is then measured by using the *string edit distance function* [5]. The *lsim* of two sets of name tokens is the average of the best similarity of each token with a token in the other set.

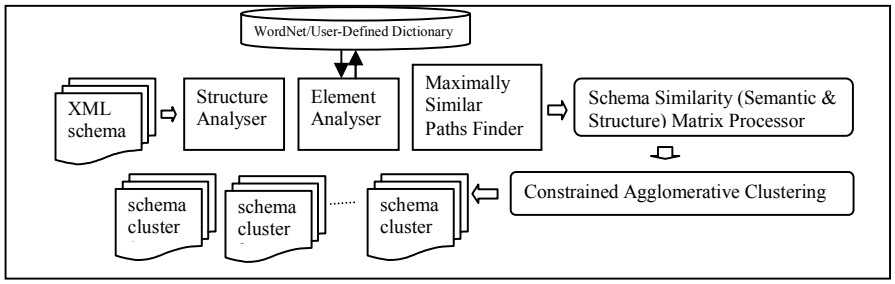


Figure 1: The architecture of XMine methodology

The ***Maximally Similar Paths Finder*** identifies the paths and elements that are common between the hierarchical structures of pairs of schemas. We adapt the sequential pattern mining algorithm [1] to infer similarity between element structures. The structure of a schema tree is represented by a set of path expressions. A path is represented by a unique sequence of element nodes following the containment links from the root to the corresponding nodes as a resultant of traversing the schema tree from root to the leaf node. A path expression is denoted as $<x_1, x_2, \ldots x_n>$ where x_1 is a root node and x_n is a leaf node. Let the set of path expressions, *PE*, in a schema tree be $\{p_1, p_2 \ldots p_n\}$. In a set of paths, a path p_j is maximal if it is not contained by any other path expression or no super path of p_j is frequent.

The overall degree of similarity based on the element and structure similarity is then computed in ***Schema Similarity Matrix Processor***. The maximal similar paths serve the basis for the element structural similarity that emphasizes on the hierarchical information of the element, which cover the context of an element defined by its ancestor (if it is not a root) and descendant elements positioned in the path expressions is measured. The element semantic similarity that involves the linguistic and constraint similarity between elements contained in the maximal large paths is also computed.

Let us assume two schemas: base schema ($schema_b$) and query schema ($schema_q$) and the corresponding base tree T_B and query tree T_Q. A unique set of path expressions are obtained by traversing both the base and query trees, denoted as PE^B and PE^Q respectively. A set of maximal similar paths (*MPE*) represents a number of common paths that exist in both base and query tree. The corresponding full path expressions that contain a *MPE* from the both PE^B and PE^Q sets are identified. The similarity coefficient of a particular maximal similar path (MPE_k), ***maxpathSim***, uses the similarity coefficient of its corresponding base and query path expressions, which refers to the path similarity coefficient, ***pathsim***.

$$maxpathSim(MPE_k) = \frac{\sum_{i=1}^{|PE^B|} \sum_{j=1}^{|PE^Q|} pathSim(PE^B_i, PE^Q_j, Threshold)}{PE^B_i \in MPE_k \quad PE^Q_j \in MPE_k}$$
$$\frac{}{Max_{PE^B \in MPE_k, PE^Q \in MPE_k} (|PE^B|, |PE^Q|)}$$

Similarity between two path expressions (***pathSim***) is computed by taking into account the similarity coefficient between the linguistic names, constraints, and path name of every element in the both PE^B_i and PE^Q_j,. This checks a one-to-one mapping of elements in the path expressions, that is an element in PE^B_i matches, at most, one element in PE^Q_j.

$$pathSim(PE^B_i, PE^Q_j, Threshold) =$$
$$\frac{\sum_{b=1}^{|PE^B_i|} \sum_{q=1}^{|PE^Q_j|} baseSim(e_b, e_q) * PNC(e_b, PE^B_i.e_1, e_q, PE^Q_j.e_1, Threshold)}{Max(|PE^B_i|, |PE^Q_j|)}$$

The linguistic and constraints similarity of the elements is derived from the base element similarity coefficient, ***basesim***, which obtained by weighted sum of linguistic similarity coefficient, ***lSim*** and constraint similarity coefficient, ***contraintSim*** of the elements:

$$baseSim(e_1, e_2) = w_1 * lSim(e_1, e_2) + w_2 * constraintSim(e_1, e_2)$$

where weights $w_1 + w_2 = 1$. The cardinality constraint coefficient, ***constraintSim*** is determined from the cardinality constraint compatible table as used in [4].

The path name coefficient, ***PNC***, measures the degree of similarity of the two element names in two given paths. *PNC* differentiates two elements with the same name but in different level position in the common paths (e.g., *book.name* and *book.author.name*) or in their context (e.g., a patient's name and a physician name). The context of an element *e* is given by the path from *root* element to an element *e,* denoted as *e.path(root)*. Thus the path from root element to element e is an element list denoted by $e.path(root) = \{root, e_{pi}, \ldots, e_{pi}, e\}$.

$$PNC = \frac{\sum baseSim}{Max(\ |dest_1.path(source_1)|, |dest_2.path(source_2)|\ )}$$

Every schema similarity value between each pair of schemas is mapped into the *schema similarity matrix*. This matrix becomes the input to the clustering process. XMine uses the ***constrained agglomerative clustering technique*** [6] to group schemas similar in structure and semantics to form a hierarchy of schema classes. The similarity between two schemas is computed by:

$$schemaSim(schema_b, schema_q) = \frac{\sum_{k=1}^{|MPE|} MaxpathSim(MPE_k)}{\max(|PE^B| + |PE^Q|)}$$

In the final phase, the discovered schema patterns are visualized as a tree of clusters called dendogram. This visualization facilitates the generalization and specialization process of the clusters to develop an appropriate schema class hierarchy. Each cluster consisting of a set of similar schemas forms a node in the hierarchy,

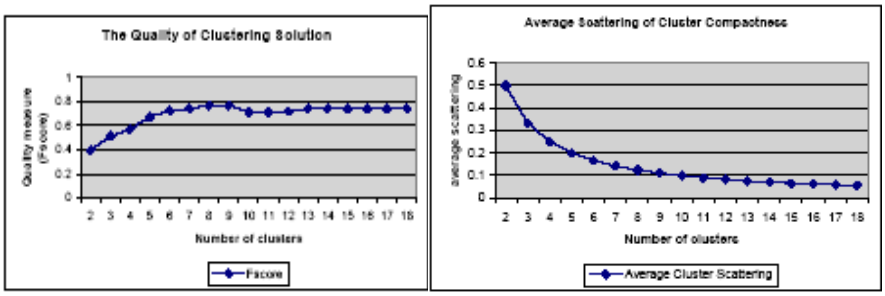
where all nodes are at the same conceptual level. Each cluster may be further decomposed into several schema sub clusters, forming a lower level of the hierarchy. Clusters may also be grouped together to form a higher level of the hierarchy.

A new schema can now be generalized. First, the schema is generalized to the identifier of the lowest subclass to which the schema belongs. The identifier of this subclass can then, in turn, be generalized to a higher-level class identifier by climbing up the class hierarchy. Similarly, a class or a subclass can be generalized to its corresponding superclasses by climbing up its associated schema class hierarchy.

3 Empirical Evaluation

The 180 schemas from various domains and sources with the nesting levels of 2-20 and nodes varying from 10 to 1000 are used in experiments. The validity and quality of the XMine clustering solutions are verified using two common evaluation methods: (1) *FScore* measure and (2) the intra-cluster and inter-cluster quality.

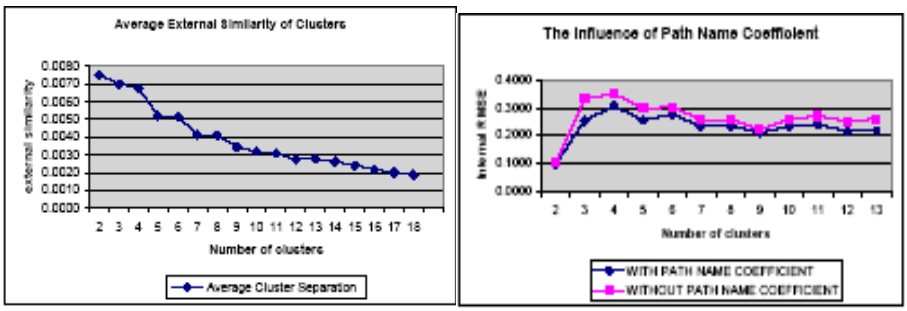
The *FScore* result of the 9-clusters solution shows the best *FScore* (figure 2). When the process reached the 13-clusters solution, the clustering quality is stabilized. The XMine maximizes the intra-class similarity by decreasing the average scattering compactness of clusters as the number of clusters increases (figure 3). This is because the greater the number of clusters specified in the solution, clusters are further decomposed into smaller subclusters containing more highly similar schemas. The figure 4 also shows that the average external similarity between clusters decreases as the number of clusters increases. As the numbers of cluster increases, a smaller size of clusters is produced consisting of highly similar schemas and hence highly dissimilar with schemas in other clusters. Based on these observations, the 13-clusters solution provides the optimal clustering model for the input data set.



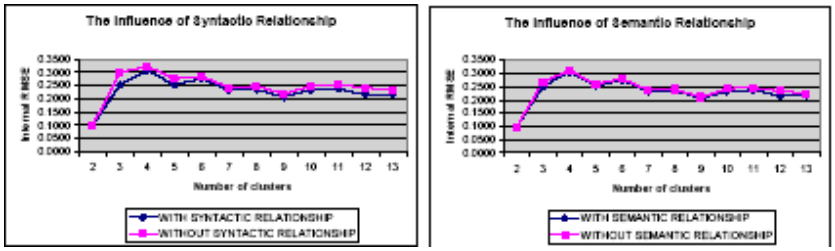
Figures 2 & 3 : FScore and Intra-class similarity Performance

XMine is also examined to test the sensitivity in computing the schema similarity coefficient (*schemaSim*). Figure 5 shows that the effect of the *PNC* on clustering is very significant. Without inclusion of *PNC*, the element names with the same semantics but occur in different position in the hierarchy path name (i.e. *book.title* and *book.author.title*) cannot be identified and discriminated. Without the semantic relationship, XMine is still able to handle the linguistic similarity between element names relatively more effectively (figure 7) than without the syntactic

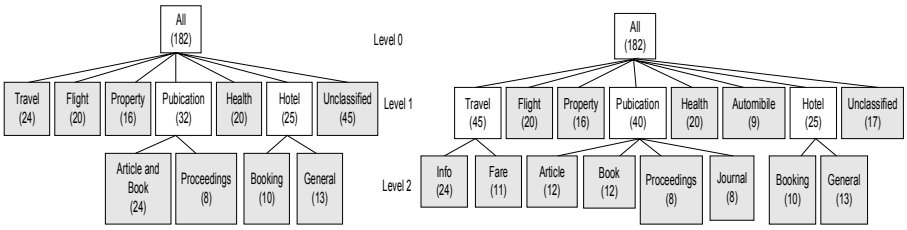
relationship (figure 6). Therefore, for what concerns element names in many cases, syntactic similarity measure could be more reliable than semantic similarity measure.



Figures 4 & 5: Inter-class similarity & Influence of Path Name coefficient



Figures 6 & 7: Effect of Syntactic and semantic relationships on clustering



Figures 8 & 9: The cluster decomposition for 9 & 13 number of clusters

Figures 8 & 9 display the clusters decomposition for 9 and 13 numbers of clusters. The shaded nodes in the hierarchy represent the actual clusters of the schemas. The unshaded nodes represent the generalization class of the low-level schema classes. Each node is labelled with the class name and the size of the class. The progression in clustering process achieves the disjoint and very specific classes of documents (i.e., lesser unclassified documents). The classes become very small in size, and may not sufficient to be considered as independent classes as they may be only holding specific schemas (as it happens in the case with 18 clusters). Generally, the clusters consist of a very small number of members called noises and outliers.

4 Conclusions

This paper presented the XMine methodology that clusters the schemas by considering both linguistic and structural information of elements in the maximal similar paths, as well as the context of an element, which is defined by its level position among other elements in the path expressions. The context of elements takes into account the elements that do not locate at the same level of the hierarchies tree, but they are similar.

The experimental evaluation shows the effectiveness of XMine in categorizing the heterogeneous schemas into relevant classes that facilitate the generalization of an appropriate schema class hierarchy. From the sensitivity evaluation, it is shown that the XMine pre-processing components highly influences the quality of clusters.

The current implementation and experiments of XMiner uses XML DTDs as schema definition language. However, XML Schema is likely to replace DTD in the future. The shift from DTDs to XML Schema is considerable straightforward with more constraint procedures to be developed in the XMine pre-processing phase for dealing with semantic extension provided in XML Schema. XMine's element analyser process can also be extended by categorizing elements into similar semantic and syntactic concepts. The purpose of element categorization is to reduce the number of element-to-element comparison. The element categorization based on their data types and linguistics content will accelerate the element comparison process by only matching elements that belong to the same element categories.

References

1. Agrawal, R., & Srikant, R. (1996). *Mining Sequential Patterns: Generalizations and Performance Improvements. Proceeding of the* 5th International Conference on Extending Database Technology (EDBT'96), France.
2. Bertino, E., Guerrini, G. & Mesiti, M. (2004). A Matching Algorithm for Measuring the Structural Similarity between an XML Document and a DTD and its applications. Information Systems, 29(1): 23-46, 2004.
3. Lee, J. W., Lee, K., & Kim, W. (2001). *Preparations for Semantics-Based XML Mining.* The 2001 IEEE International Conference on Data Mining (ICDM'01), Silicon Valley, CA.
4. Lee, L. M., Yang, L.H., Hsu, W., & Yang, X. (2002). *XClust: Clustering XML Schemas for Effective Integration.* The 11th ACM International Conference on Information and Knowledge Management (CIKM'02), Virginia.
5. Rice, S. V., Bunke, H., & Nartker, T.A. (1997). Classes of Cost Functions for String Edit Distance. *Algorithmica, 18*(2), 271-280.
6. Zhao, Y., & Karypis, G. (2002, November 4-9, 2002). *Evaluation of Hierarchical Clustering Algorithms for Document Datasets.* The 2002 ACM CIKM International Conference on Information and Knowledge Management, USA.